Ontology-Based Methods for Enhancing Autonomous Vehicle Path Planning

Ron Provine[1], Craig Schlenoff[2], Stephen Balakirsky[2], Scott Smith[1], Mike Uschold[1]

[1]Boeing Phantom Works
P.O. Box 3707, m/s 7L-43
Seattle, WA 98124-2207, USA
ronald.c.provine@boeing.com

[2]Nat. Inst. of Standards and Technology
100 Bureau Drive, Stop 8230
Gaithersburg, MD 20899, USA
{stephen, craig.schlenof}@nist.gov

Abstract

We report the results of a first implementation demonstrating the use of an ontology to support reasoning about obstacles to improve the capabilities and performance of on-board route planning for autonomous vehicles. This is part of an overall effort to evaluate the performance of ontologies in different components of an autonomous vehicle within the 4D/RCS system architecture developed at NIST. Our initial focus has been on simple roadway driving scenarios where the controlled vehicle encounters potential obstacles in its path. As reported elsewhere [9], our approach is to develop an ontology of objects in the environment, in conjunction with rules for estimating the damage that would be incurred by collisions with different objects in different situations. Automated reasoning is used to estimate collision damage; this information is fed to the route planner to help it decide whether to plan to avoid the object. We describe the results of the first implementation that integrates the ontology, the reasoner and the planner. We describe our insights and lessons learned and discuss resulting changes to our approach.

1 Introduction

a) Statement of the Problem

An autonomous vehicle is an embodied intelligent system that can operate independently from human supervision. The field of autonomous vehicles is continuing to gain traction with both researchers and practitioners. Funding for research in this area has continued to grow over the past few years, and recent high profile defense-related funding opportunities have started to push theoretical research efforts into practical use.

To behave appropriately in an uncertain environment, many researchers and practitioners believe that "the vehicle must have an internal representation (world model) of what it experiences as it perceives entities, events, and situations in the world. It must have an internal model that captures the richness of what it knows and learns, and a mechanism for computing values and priorities that enables it to decide what it wishes to do." [3]. The inability to accurately model the world hinders effective task planning and execution and thus the overall effectiveness of the vehicle. A major challenge in autonomous vehicles is the ability to accurately maintain this internal representation of pertinent information about the environment in which the vehicle operates. Our approach is to enhance existing world modeling methods using an ontology-based model to represent certain aspects of the vehicle's environment.

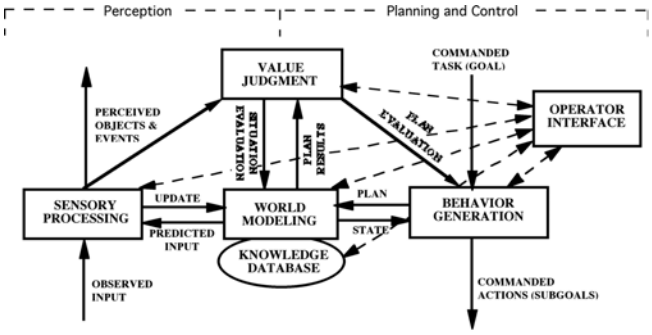

Figure 1: RCS Node

b) The 4D/RCS Reference Model Architecture

For reasons discussed more fully in [9] we have selected the Real-Time Control System (4D/RCS) [1,2] as the architecture in which we implement and evaluate the use of ontologies for autonomous vehicles. 4D/RCS is a hierarchical, distributed, real-time control system architecture that provides clear interfaces and roles for a variety of functional elements.

Under 4D/RCS, the functional elements of an intelligent system can be broadly considered to include: behavior generation (task decomposition and control), sensory processing (filtering, detection, recognition, and grouping), world modeling (store and retrieve knowledge and predict future states), and value judgment (compute cost, benefit, importance, and uncertainty). These are supported by a knowledge database and a communication system that interconnects the functional elements and the knowledge database. This collection of modules and their interconnections make up a generic node in the 4D/RCS reference model architecture (see Figure 1) [3]. A generic node is defined as a part of the 4D/RCS system that processes sensory information, computes values, maintains a world model, generates predictions, formulates plans, and executes tasks. Each module in the node may have an operator interface.

c) Vehicle Level Planning Within 4D/RCS

Planning is done at every level within the 4D/RCS architecture. In this effort, we will be focusing on vehicle-level planning, which plans approximately 20 seconds into the future with a replanning rate of one to two seconds.

As described in [5], the NIST vehicle-level behavior generation system utilizes incrementally created planning graphs to formulate potential vehicle trajectories. It combines both logic-based and cost-based planning approaches in order to allow for the creation of logic-constrained cost optimal plans with respect to possibly dynamic environments, user objectives, and constraints. It includes the ability to implement both hard and soft constraints. Hard constraints are based on derived domain feature predicates and are used to incrementally construct a planning graph. Soft constraints allow the system to exhibit a preference for one form of state transition over another. These preferences are controlled through the use of a cost function during the incremental construction of the planning graph and lead to favoring certain system behaviors over others.

d) Our Initial Focus

An ontology component promises to be helpful in many aspects of the 4D/RCS architecture. For our purposes, an ontology is a formal, declarative, and computer-interpretable knowledge representation that supports automated reasoning to infer additional information and check constraints.

Our initial efforts are aimed at assisting the planner in deciding upon the most cost-effective plan, focusing on the value judgment and behavior generation components, in particular. The planner utilizes the results of reasoning over the ontology.

A major assumption in this work is that objects in the environment will be identified by lower level sensory processing algorithms by the time the vehicle-planner operates on them. This is a major research area in itself, but object classification is outside the scope of this paper.

The value judgment component evaluates perceived and planned situations. It computes what is important (for attention), and what is rewarding or punishing (for learning). The value judgment component assigns priorities and computes the level of resources to be allocated to tasks. It assigns values and costs to recognized objects and events, and computes confidence factors for observed, estimated, and predicted attributes and states [2]. The outputs of the value judgment component are used by the behavior generation component to select and set priorities during route planning.

Our approach is to use ontology-based reasoning to better inform the planner about the costs and consequences of colliding with other objects. By representing the factors that could impact a path's cost, the ontology is used to reason over the information that is available to determine what the consequences of a collision would be. Further reasoning is then performed to determine the cost of these consequences. This cost is fed back to the planner for consideration when deciding the "cheapest" plan for the system to execute. In cost-based planners, the cheapest plan is considered the best plan.

In the next section, we describe the details of how we integrated the ontology with the planning system. The issues and insights developed during this first phase of the project are described in Section 3. We conclude and consider future work in Section 4.

2 Integrating An Ontology With A Path Planner

a) Scenario

In its full generality, the problem of automated vehicle path planning is extremely challenging. We limit our focus to roadway driving, and the objects and obstacles that might likely be encountered. We start with the simple scenario illustrated in Figure 2. The autonomous vehicle (black rectangle) is in the left lane of a four-lane two-way undivided highway. An object is detected in our lane. The goal is to formulate an optimal route plan that takes into account the estimated damage from a collision with the object. In this first

implementation phase, the main role of the ontology component is to provide assessments of collision damage.

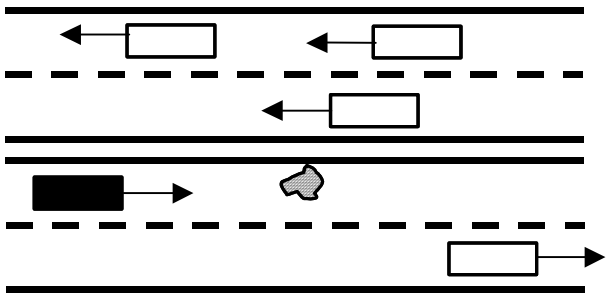

Figure 2: Simple Driving Scenario

b) Constraints

Autonomous vehicle path planning places a number of requirements on the ontology and the tool in which it is implemented.

- Our implementation platform – an autonomous vehicle - requires real-time performance by the nodes. Since the 4D/RCS architecture is hierarchical, the definition of real-time will change with level. At the level in which we are planning to apply the ontology (the vehicle level), planning decisions must be made within two seconds.

- In the 4D/RCS architecture, the behavior generation component makes calls to the value judgment component for each node in the plan graph that is generated by the cost-based planner. As there may be thousands of nodes in a plan, any ontology reasoning engine operating within the value judgment module must support high transaction rates.

- There are concepts in the ontology that express relationships between measured properties, e.g. the closing velocity of the vehicle and object and the ratio of their masses. This necessitates the ability to represent and reason about (i.e. do computation with) real, continuous-valued variables.

- The nature of the problem is such that the system performance must degrade smoothly. Conceptually this means that the ontology must include general (default) reasoning in addition to support for specific situations.

In our initial implementation, we have met some but not all of these requirements; this is discussed below.

c) Overview of Ontology & Reasoning

The existing planner treats all objects as obstacles, and uniformly plans to avoid all of them. The ontology and reasoning component exploits the fact that only some objects are obstacles that need to be avoided. For our initial proof-of-concept experiments, we created a small ontology with a few rules that can determine the extent to which a given object may constitute an obstacle in a given situation.

The ontology contains a couple dozen objects ranging from street signs to traffic barrels to bricks and newspapers; most have very different characteristics. We also created four kinds of [potentially autonomous] vehicles: a van, a truck, a Hummer, and a motorcycle. Again these have widely varying characteristics. Each of the object classes in the ontology has a small set of characteristics associated with them, focusing on the object's rigidity, movability, crushability, and dimensions.

The attribute values for both the objects and the vehicle types were stored in a qualitative way. For example, we stated in the ontology that a fire hydrant has high rigidity and low movability (since it is a fixed to the ground).

We also describe the damage classifications for the collisions in qualitative terms. The vehicle damage classification can be one of five enumerated values: *none*, *minor*, *moderate*, *major*, and *catastrophic*.

Based upon the objects' and the vehicles' characteristics, we developed a set of rules that described how to combine various sets of characteristics to determine a qualitative cost for the collision. The first rule below states that if a situation exists in which a truck collides with a brick, this results in a situation whereby there is no damage to the vehicle. The second rule states that if a car or a van collides into a rigid fixed object (such as a fire hydrant), then there is catastrophic damage to the vehicle.

```
;;; Truck + Brick = None.
 (implies
       (and situation
              (some has-vehicle truck)
              (some has-potential-obstacle
                    brick))
       sit-VD-None)

;;;(Car or Van)+Rigid-Fixed-Object=Catastrophic.
 (implies
       (and situation
              (some has-vehicle (or car van))
              (some has-potential-obstacle
                    rigid-fixed-object))
       sit-VD-Catastrophic)
```

See [8], for more details on the ontology and the knowledge representation issues. Next, we consider some of the implementation details.

d) Implementation Details

We cast the problem of assessing collision damage as one of classification into one of several pre-existing categories (from none to catastrophic), as noted above. The use of description logic [4] is an obvious choice for classification reasoning. Description logic reasoners are finely tuned for performing classification, and thus are very fast for the type of reasoning we require. This is important because the planner needs to query the ontology component up to a few thousand times a second to get damage estimates for the many nodes being explored in the search space. Another important advantage of using description logic is automated semantic consistency checking to ensure that there are no logical errors in the ontology. For example, when creating an ontology about farm animals, one might say the following: 1) cows & sheep are animals; 2) cows are vegetarians; 3) vegetarians eat no animals, and eat no parts of any animal; 4) mad cows eat the brains of sheep, and 5) a mad cow is a kind of cow. However, there is an inconsistency lurking: a mad cow can't be a kind of cow because that means it is a vegetarian that eats sheep brains.[1] The system can detect inconsistencies such as these that can only be explained by chains of reasoning steps. Automated detection of such errors greatly increases confidence that the ontology is correct.

We started out by using the ontology editing tool, OilEd [6] to construct our ontology, and the FaCT inference engine[2] to 1) check for semantic consistency and 2) perform the damage assessment reasoning. FaCT is directly connected to OilEd, so consistency checking is done by pressing a button. Various minor semantic bugs were identified and manually fixed. This works much like a type checker in a programming environment. Using the OilEd/FaCT combination, we identified a variety of possible situations, simulating the reasoning that would eventually be done on-board the vehicle, to ensure that the results were correct.

For the purpose of this implementation, we assumed that all objects in the environment were fully recognized. As such, the values of the object's attributes were be stored *a priori* in the ontology and compared with the data coming from the autonomous vehicle's planning system. When a vehicle encountered an object that was represented in the ontology, the values of the attributes of that object were used to help determine the cost of collision. For example, when the vehicle encountered a traffic barrel, the traffic barrel object represented in the ontology was accessed and the values of the attributes that were predefined for a traffic barrel were utilized.

The next step was to integrate the FaCT inference engine running as a server with the planner using function calls from the planner to the server. The planner required a C++ interface which FaCT lacked; instead, we chose an alternative description logic inference engine: RACER [8].

Connecting RACER to the planner was fairly straightforward. We did encounter some difficulties in converting the ontology to a format suitable for uploading into RACER. Although OilEd exports to a format that RACER should be able to import, there were some compatibility problems. For expedience in getting the demonstration working, we manually encoded portions of the OilEd ontology into a Lisp syntax suitable for RACER. During this process, we made certain enhancements and additions to the ontology. At this temporary stage in development, we now have no ontology editing tool that we can use to view and maintain the ontology. This reflects the relatively immature support for interoperability that exists in today's ontology tools.

With an enhanced ontology, we connected RACER to the planner. When the planner requires a collision damage assessment, the planner passes to RACER the type of vehicle it is and the type of object it encountered in the environment. Using classification rules, the inference procedure classifies the current situation into one of the pre-determined damage categories. The rules map certain types of vehicles and objects, based on their characteristics, to the damage categories. For example, the situation called: "sit-VD-Catastrophic" is the name of the situation which is defined to be any whose `hasVehicleDamage` attribute is equal to "`CS-Catastrophic`"[3]. The qualitative damage category is passed back to the planner, which converts it to a numeric cost suitable for use in its numeric search algorithms.

For our proof-of-concept demonstration, the planner is connected to a driving simulation package. We intend to connect this up to a real vehicle in the near to medium term.

Experimental Results

We performed four tests with the integrated planner and ontology. In all cases, the planner evaluated a path

[1] Example is from the OilEd/FaCT download.
[2] See: http://www.cs.man.ac.uk/~horrocks/FaCT/
[3] CS is for Cost Severity; VD is for Vehicle Damage.

in which the vehicle, driving on a two-lane, one-way roadway, encountered an object.

Case 1: Vehicle: Hummer Object: Brick

In this case, the Hummer ran over the brick. The ontology and reasoner determined that a high rigidity, large vehicle striking a high density, small object would cause no damage.

Case 2: Vehicle: Hummer Object: Traffic Barrel

In this case, the Hummer changed lanes to avoid the barrel. The ontology and reasoner determined that a high rigidity, large vehicle striking a high density, medium-sized object would cause significant enough damage to justify the extra cost of changing lanes.

Case 3: Vehicle: Motorcycle Object: Brick

In this case, the motorcycle changed lanes to avoid the brick. The ontology and reasoner determined that a medium rigidity, medium-sized vehicle striking a high density, small object would cause significant enough damage to justify the extra cost of changing lanes.

Case 4: Vehicle: Motorcycle Object: Traffic Barrel

In this case, the motorcycle changed lanes to avoid the barrel. The ontology and reasoner determined that a medium rigidity, medium-sized vehicle striking a high density, medium-sized object would cause significant enough damage to justify the extra cost of changing lanes.

The response time of the ontology for the purpose of planning at the vehicle level seemed to be sufficient considering the timing constraints placed on the planner, though more trials would need to be performed to verify this.

3 Observations

This first experiment showed that the integration of a planner with an ontology was not only possible, but also improves the decision that the planner made in the presence of certain objects. Without the ontology integrated, the planner avoids all obstacles, independent of their type. While this is the most conservative approach to driving, it also causes the vehicle to perform unnecessary lane changes, which puts the vehicle in more jeopardy than necessary.

However this was just the first step. In this section, we reflect on the choices we made, indicating what worked well and where changes and improvements are necessary in the future. Many things were not taken into account during this initial exercise that would need to be in a fielded version of the ontology. They include the vehicle's speed, the vehicle's condition before striking the obstacle, additional qualitative characteristics of the object and the vehicle, and the mission the vehicle is performing. These issues, among others, are discussed in more detail in this section. For each we indicate what has been done already, vs. what is planned for future work.

a) Obstacles as Roles

An initial examination of the driving scenario led to the recognition that an obstacle is a role that an object plays in a certain situation. A person walking along the sidewalk is not an obstacle to a vehicle on the road; however that same person in the same location dashing toward the road to get a ball is an obstacle to be avoided. A general theory of obstacles should define a set of conditions that determine whether an object is an obstacle. That determination depends on the *relationship* between the object and another entity (for us, an autonomous vehicle). If the relationship entails impeding the progress of the vehicle, or impeding the vehicle's ability to carry out its goals, then the object is an obstacle.

Recognizing the situation-dependent nature of obstacles, we created a generic notion called **Situation** for collecting information that is relevant for determining whether objects are obstacles that must be avoided. This has proven to be an excellent choice.

In our initial implementation, each situation has an associated autonomous vehicle and a potential obstacle. In turn, the vehicles and obstacles have associated properties such as rigidity, weight and density that are used to determine potential collision damage. In future implementations, we will add other important factors such as driving speed, road conditions, vehicle clearance, object height, etc.

b) Colliding With Objects Incurs a Cost

As mentioned previously, we are using ontology reasoning to assist the planner in determining possible collision damage. Operationally, we provide inputs to the value judgment module (Figure 1) for use in computing costs for plan segments. This led us immediately to view collisions with obstacles as cost factors. In this formulation, colliding with a rigid fixed object (e.g. a large cinder block) results in a higher cost than colliding with a small crushable object (e.g. a paper on the road). No damage corresponds to zero

cost, in which case the object is not an obstacle for that situation[4].

This approach seemed both intuitive and easy to integrate into a value judgment calculation. However, examination of several simple scenarios led to the realization that constructing these costs would not be straightforward. Consider the following situations:

- You are driving by yourself in heavy traffic when you encounter a brick in your lane.
- You are carrying a piece of sensitive, fragile equipment when you encounter a brick in your lane.
- You are delivering urgent medical supplies when you encounter a brick in your lane.
- You are delivering urgent medical supplies when you encounter a small child in your lane.

Discussion of these scenarios led us to recognize that damage to the vehicle is not the only thing to consider in evaluating a potential collision. We must also take into account the amount of damage incurred by the payload, as well as the damage to the potential obstacle. More conservative driving is called for if the payload is both fragile and valuable. Also, a vehicle carrying urgent medical supplies may risk damage to the vehicle by running over a brick, but not by running over a small child. Hence **Payload** information must be included in the Situation description, in addition to information about the vehicle and the potential obstacle. Note that the potential obstacle is used in two ways to assess damage: the damage to the object itself, and the damage to the vehicle resulting from a collision with the object.

In the initial implementation, we have three situation attributes for characterizing collision damage: vehicle damage, object damage and payload damage. The planner converts the qualitative damage assessment categories (from None to Catastrophic) into numeric costs.

c) The Situation must consider the Mission

The above scenarios also led to the identification of the **Mission** as an important factor in determining cost. This is expressed in terms of the relative importance placed on maintaining or restricting the values for the integrity of the vehicle, the payload and the obstacle.

If the mission is to get from point A to point B as quickly as possible, regardless of the resulting condition of the vehicle or the payload, one can visualize the typical Hollywood car chase – nothing is an obstacle. In our formalization, for this mission all obstacles would have zero cost.

However, in a more typical mission, such as commuting to work, there is a desire to minimize damage to the vehicle and the payload (the driver and passengers). In this situation, most obstacles would have a significant cost.

We have not yet begun exploring approaches for representing the mission requirements in our ontology.

d) Damage Costs must be Accumulated

Our original formulation only considered the estimated damage from a single collision; we [implicitly] assumed that the vehicle [or object or payload] was already in perfect condition. We did not distinguish between the incremental damage due to a particular collision, and the overall integrity of the vehicle, which could have suffered prior collisions. For example, a vehicle might have a wheel that is close to falling off, so that a collision that would be inconsequential for a new vehicle would be significant for this one. One can also envision scenarios (e.g. driving on a fresh gravel road) where repeated minor collisions eventually result in major damage. What is important to a navigation decision is not the incremental damage to a given collision, but rather the overall *integrity* of the vehicle, object, or payload that would result from that collision. The latter takes into account the pre-collision integrity as well as the incremental damage due to the current collision.

Integrity is a property of Object, and qualitatively describes the condition of that object with respect to the amount of damage it has accumulated. After a collision, the object integrity might remain the same, or decrease. Note that integrity is inversely related to the accumulated damage to the object. So an accumulated damage category called "None" corresponds to the highest level of integrity, and "Catastrophic" means the integrity has vanished. Accumulation of damage has been partially implemented.

Limitations of description logics prevented a more complete implementation and necessitate a move to different ontology tools. Anticipating ontology tools that can do the required reasoning, we have created a knowledge base of rules that map the current integrity value to the new value after a collision for each of: Vehicle, Obstacle and Payload. Thus, for the scenario where a vehicle collides with a melon, the vehicle and payload integrity would be unchanged, but the melon's integrity would vanish; it would be destroyed. An example of a more general rule is summarized in the table below.

[4] An alternate view is that all objects are obstacles and zero cost corresponds to the degenerate case of an obstacle.

Initial Damage	Final Damage
None	Moderate
Minor	Moderate
Moderate	Severe
Severe	Catastrophic
Catastrophic	Catastrophic (unchanged)

The right column indicates the resulting damage to the vehicle, payload or obstacle given the initial damage in the left column, and that the incremental damage due the the current collision is moderate.

e) **Obstacles have Qualitative Characteristics (which is what we <u>really</u> care about)**

To determine the likely cost of a collision with an object, you need to know certain characteristics of the object. An empty cardboard box in the middle of the road will cause minimal damage to a vehicle that strikes it, compared to if the box contains a television inside it. The more that is known about the characteristics of the object, the better an approximation can be made about the implications of colliding with it. Identifying an object's characteristics can help identify the object. Also, if an object has been identified, then the knowledge base can be consulted to determine other characteristics of the object that may not be visible (e.g., rigidity, elasticity, etc.). The key oint is that damage determination is still based upon the characteristics of the objects as opposed to the object identification.

Even though a physics-based approach could provide very detailed information about the damage that could be caused by colliding with obstacles in the environment, this is overkill for planning done at the vehicle level. At this level, the planner does not care about the details of momentum transfer or impact calculations; it simply wants a rough classification of what type of damage could be expected if collision were to occur.

This is very similar to what human drivers do. They see an object in the road that they may or may not recognize, and make a quick determination of what type of damage would be expected if they were to run into this obstacle. There is no physics involved in this; it is simply a high-level determination.

In our research, we have identified a set of characteristics that appear to have the strongest impact on determining the damage that can be incurred from collision. They are:

- Rigidity
- Density
- Weight
- Crushability
- Movability
- Dimensions (height, transverse length, longitudinal length)
- Sharpness
- Shatterability/Breakability
- Elasticity

The initial implementation contains the first five of the above characteristics. For each, quantitative descriptions are used (e.g., high, medium, low) to describe their value. This information is determined '*a priori'* (e.g., I know that a rubber ball is very crushable and has low density). The above list is not meant to be exhaustive; it will grow as the project progresses.

In future implementations, we hope to populate the qualitative characteristics using sensor data. Based on the sensor data, rules would be fired to match perceived characteristics of the object to qualitative descriptions. When a characteristic of the object cannot be determined via sensor data (e.g. shatterability), this information would be marked as unknown and the application implementing the ontology would determine how to handle it.

f) Dimensions, Orientation and Calculation

In working through the qualitative physics that influence driving decisions (and determine the integrity transformations), we realized an initial evaluation is performed to determine which of three subclasses of "collisions" will occur:

- the vehicle can avoid the obstacle, swerving around it while remaining in its lane,
- the vehicle can pass over it, adjusting its path so that the obstacle passes cleanly underneath, or
- the vehicle will run into the obstacle.

Note that changing lanes and avoiding the obstacle are not on this list; only the situations where the vehicle and the object occupy the same space are considered.

Thus, a brick at the edge of the lane need not be considered as an obstacle (i.e., it has zero cost). Similarly, a long board lying across the road will have some non-zero cost, while the same board lying along the lane will have a zero (or nearly zero) cost because it is easily straddled.

These considerations require that we represent the dimensions of obstacles in our ontology, as well as some of the basic specifications of the vehicle (e.g. wheel base and ground clearance) and the relative orientation and position of the vehicle and obstacle. It also requires that the ontology tool have the capability for performing mathematical operations and comparisons.

Pursuing the development of the qualitative physics, we found that the closing velocity and the relative masses of the Situation participants, both real-valued quantities, were necessary to produce reasonable cost values.

In the initial implementation relative masses objects were considered. Orientation, velocity and various computations will be included in future versions.

g) Description Logic is Limiting

From our initial tests, it is clear that there are limits to using a description logic ontology language and reasoner for our task. For example, we cannot return the value of the vehicle or payload integrity (as in a function call) and indicate it as either unchanged or that it has been incremented by some level in the damage severity scale. There is no facility for doing arithmetic (although arithmetic comparison such as greater-than is possible). We might need to compute the ratio of the weights of the potential obstacle and the vehicle (if it was sufficiently high, then the obstacle will cause at worst, minor damage). A DL is also very limited in the kinds of rules that it can express. Similar problems were discovered in an attempt to use description logic classification to implement a semantic publish and subscribe system [10].

We are currently exploring different rule languages. In the absence of a decision of which to use, we have created a revised, rationalized and extended ontology in a home-grown Prolog-like syntax, for eventual encoding into a working system.

4 Future Work and Conclusion

The overall goal of this work is to apply ontologies to enhance the capabilities and performance of autonomous vehicles, particularly in the area of path planning. In order to do this, we are initially using an ontology to determine the damage resulting from collisions between autonomous vehicles and different types of objects that could be encountered during on-road driving.

Although our initial experiment showed promise, both the ontology and the experiment lacked a number of fundamental concepts that would be necessary for the robustness of this approach could be proven. Concepts such as the vehicle's speed, the vehicle's integrity before striking the obstacle, additional qualitative characteristics of the object and the vehicle, and the mission the vehicle is performing will be added in future versions of the ontology and reasoner.

In addition, due to the limitations posed by description logics, as indicated in this paper, the rules will be ported over to CLIPS and the ontology will be developed in Protégé. CLIPS was chosen due to its real-time capabilities and its ability to perform numeric computations [11]. Protégé was chosen due to its ability to export an ontology into CLIPS format, its ease-of-use, and its strong user community. Very initial efforts in using CLIPS and Protégé within this effort have confirmed the appropriateness of these tools for our purposes.

As part of our reassessment, we are also attempting to address a question that we posed for ourselves in [9]:

- To what extent can a general theory of obstacles be adapted to a wide variety of autonomous vehicle applications? Can we have a single ontology for multiple types of vehicles and contexts? How much will they have to be tailored? This is analogous to the long-time question about standard upper ontologies (SUO), but within a limited domain. Can there be a SUO of obstacles?

Guarino [7] suggests the concept of defining different kinds of ontologies according to their level of generality. We have developed a suggested decomposition of the obstacle ontology into top-level, domain, task and application ontologies using this approach, as follows:

- Top-Level Ontologies
  - Physical Objects
- Domain Ontology
  - Vehicles
  - Payloads
  - Road Segments (future)
  - Rules of the road (future)
- Task Ontology
  - Classification
  - Mission
- Application Ontology
  - Vehicle Integrity Transformations (future)
  - Obstacle Integrity Transformations (future)
  - Payload Transformations (future)
  - Situation Classification
  - Obstacles

From this decomposition, we suggest that the top-level, domain and task ontologies can be adapted to a wide variety of autonomous vehicle applications. Demonstration of this result will depend, however, on successfully resolving the issues that we have identified with our current implementation.

References

1. Albus, J., "Outline for a Theory of Intelligence," *IEEE Transactions on Systems Man and Cybernetics*, Vol. 21, 1991, pp. 473-509.

2. Albus, J. and et.al., "4D/RCS Version 2.0: A Reference Model Architecture for Unmanned Vehicle Systems," NISTIR 6910, National Institute of Standards and Technology, Gaithersburg, MD, 2002.

3. Albus, J. and Meystel, A., *Engineering of Mind*, John Wiley & Sons, Inc. 2001.

4. Baader, F., McGuinness, D., Nardi, D., and Patel-Schnedier, F., *Description Logic Handbook: Theory, Implementation and Application*, Cambridge University Press 2002.

5. Balakirsky, S. and Herzog, O., "Planning with Incrementally Created Graphs," NIST, 6895, Gaithersburg, MD, 2002.

6. Bechhofer, S., Horrocks, I., Goble, C., and Stevens, R., "OilEd: a reason-able ontology for the semantic web," *Proc. of the Joint German Austrian Conference on AI, number 2174 in Lecture Notes In Artificial Intelligence*, Springer-Verlag, 2001, pp. 396-408.

7. Guarino, N. and Welty, C., "A Formal Ontology of Properties," *LADSEB/CNR Technical Report 01/2000*, 2000.

8. Haarslev, V. and Moller, R., "RACER System Description," *Proceedings of the First International Joint Conference on Automation Reasoning (IJCAR'01), number 2083 in Lecture Notes in Artificial Intelligence*, Springer-Verlag, 2001, pp. 701-705.

9. Schlenoff, C., Balakirsky, S., Uschold, M., Provine, R., and Smith, S., "Using Ontologies to Aid in Navigation Plannig in Autonomous Vehicles," *to appear in the Special Issue on Ontologies and Distributed Systems in the Knowledge Engineering Review*, 2004.

10. Uschold, M., Clark, P., Dickey, F., Fung, C., Smith, S., Uczekaj S., Wilke, M., Bechhofer, S., and Horrocks, I., "A semantic infosphere," *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, 2003.

11. Zimmerman, N., Schlenoff, C., and Balakirsky, S., "Implementing a Rule-based System to Represent Decision Criteria for On-Road Autonomous Navigation," *Proceedings of the 2004 AAAI Spring Symposium on Knowledge Representation and Ontologies for Autonomous Systems*, 2004.